A Tale of Two Failed Programs

J. Benslay, MITRE

SSTC 2010

Disclaimer: The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author."

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
APR 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Tale of Two Failed Programs	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **MITRE Corporation,202 Burlington Road,Bedford,MA,01730-1420**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at the 22nd Systems and Software Technology Conference (SSTC), 26-29 April 2010, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**20**	

Fundamental Premise

- Every reasonably complex information exchange system must have some level of systems engineering rigor and oversight – but the essential question is how much.
  - Too much rigor and oversight – and the project costs and schedule balloon while negatively affecting creativity and innovation.
  - Too little rigor and oversight – and the project may suffer from poor engineering quality, may not adhere to standards, and may not work well within established architectures.

Overview

- Tell the stories of two failed programs
  - Program Sigma
  - Program Delta Prime
- Examine the commonality of their failures
- Was it avoidable?
- What could have been done to avert the failures?
- Conclusions

Program “Sigma”

- A grand new vision for an existing capability
- Modernization of an older architecture
- Pulling together of disparate pieces
- Transition to services-based architecture and web-based thin clients
- Programmatically controlled from central group, with major sections “contracted” to other organizations for development
- Would become the exemplar of programs to emulate

Program “Delta Prime”

- The supercharging of an old capability
- Upset the status quo by circumventing time-intensive processes
- Innovate – innovate -- innovate
- Meet continuing user demand by rapidly producing new capabilities
- No goal is too far to reach
- Circumvent non-materiel problems with materiel work-arounds
- From prototype to Program of Record in very short time

“Sigma” - Beginnings

- Began as a natural evolution from an existing capability
- It would follow well-established procedures for providing and/or participating in the proposal, the plan, the requirements, the funding, the teams, the architecture, the management, the production, the testing, the security, the ….
- Significant piece of systems engineering that would lay a firm but extensible foundation for many years to come

“Delta Prime” - Beginnings

- Began as a small project by a subset of the greater user group that was frustrated with the existing system
- Did not follow traditional acquisition steps
- Prototype built under FFRDC; provided the users almost real-time software solutions
- Government oversight minimal
- Cost-plus funding scenario
- System engineering not a priority; almost an afterthought
- Requirements not tracked or captured accurately

“Sigma” - Challenges

- Scope of project was huge – conventional wisdom was to nail down the architecture first so that all participating orgs would know what to build to
- Many engineers involved and many opinions – but no one at the helm who had implemented this same kind of technology on this scale before
- Complex systems engineering process that not everyone understood
- Pressure was high from senior leadership – with frequent briefings required

“Delta Prime” - Challenges

- Prototype system demonstrated to leadership, then becomes an “Operational Prototype”
- Resistance to any changes that would trade producing new capabilities for documentation, engineering, etc.
- True users not aware of decisions; representative not consulting or informing users of the facts
- Delta Prime sponsors squared off with existing program sponsors and basically threw down the gauntlet….existing program days were numbered…..

“Sigma” - Warning Signs

- Large cast of engineers & managers had significant difficulty agreeing on fundamentals
- Early high-level concepts were not translating well to real-world practicalities (functional and security testing of SOA components)
- Process frustration (complex and not well understood) led to ad-hoc changes
- Significant change in priorities
- (Unrealistic?) expectations that the new system would be significantly faster….

“Delta Prime” - Warning Signs

- Program was required to recompete the developer contract
  - Original developer lost – inexperienced team won
- Examination of all aspects of the program revealed breadth vs depth – many features only work partially, and overall is fragile
- Funding for fixes in a version release double the cost of development of release
- Revelation that the system required “hands-on” attention in order to remain stable

“Sigma” - Critical

- Some significant program partners decided to start implementing whatever they wanted, regardless of what had been agreed-to beforehand
- Demand from leadership for demonstrable prototype that was not part of the original plan
- Key personnel turn-over
- Significant doubts at senior leadership level that technology risks could be managed
- Senior level Tiger Team tasked to determine way ahead – failed to produce consensus

“Delta Prime” - Critical

- Failure of software under new developer; causes of failure complex and difficult to address quickly
- User conferences failed to meet goals due to software issues; wasted time and money for users
- External and objective engineering looks at software revealed multiple root issues; months to address with band-aid measures
- Weekly briefs to Pentagon leadership to monitor status took key people away from addressing issues with software and focus became briefs

“Sigma” - Termination

- Highly complex set of requirements
- Objective system designs that would only provide partial replacement of existing system functionality
- Inability to move past major FAR milestone
- 40% budget cut in one cycle
- Inability to provide workable demonstration

“Delta Prime” - Termination

- Largest and most vocal user group announcement that they would use other software for next user conference
- Realization that technical fixes to several root problems would not fully address the “ilities”
- Modernization of the prototype estimated to cost more than starting from scratch
- Original developer of the prototype would not team with other developers and made unreasonable demands upon the government for a contract to fix the prototype

Commonality?

- Is there a commonality in the failure of these two very different systems?

- ***Both suffered from an inappropriate amount of systems engineering rigor and oversight.***
  - ***Sigma – too much***
  - ***Delta Prime – too little***

Avoidable?

- Should have realized that their situations weren't all that unique – and sought to apply what others had learned in similar situations
- Agile Software Development introduced in 2001 – and has some valuable principles that can be applied here.

agilemanifesto.org

Avoidable?

- Sigma: introduce a little agility into their process:
  - Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
  - Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
  - Working software is the primary measure of progress.
  - Simplicity – the art of maximizing the amount of work not done – is essential.
  - At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Avoidable?

- Delta Prime: introduce a little agility into their process:
  - Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
  - Business people and developers must work together daily throughout the project.
  - Working software is the primary measure of progress.
  - Continuous attention to technical excellence and good design enhances agility.
  - At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Conclusions

- Every IT program will face its share of challenges and difficulties
- Most of those challenges don't come from a particular technology – rather from a misapplication of methodology
- Managers and engineers must take care to continuously evaluate whether they are applying the most appropriate lifecycle methodologies, and be prepared to adjust as necessary